TEAMCENTER

Teamcenter AI Services – Local Infrastructure Setup

December 2024

SIEMENS

About Siemens Digital Industries Software

Siemens Digital Industries Software is a leading global provider of product life cycle management (PLM) software and services with 7 million licensed seats and 71,000 customers worldwide. Headquartered in Plano, Texas, Siemens Digital Industries Software works collaboratively with companies to deliver open solutions that help them turn more ideas into successful products. For more information on Siemens Digital Industries Software products and services, visit www.siemens.com/plm.

Support Center: support.sw.siemens.com
Send Feedback on Documentation: support.sw.siemens.com/doc_feedback_form

Table of Contents

This document provides steps to set up a local solution to use Teamcenter AI Chat:

OpenSearch Installation

Provides access to OpenSearch Vector Database.

OpenSearch Dashboard Installation

Allows access to create indexes and OpenSearch Customization

Creation/Modification of Users and Roles in OpenSearch

(Optional) Provides steps to make changes to configurations you have made along with the ability to create roles and users if necessary.

Download, Installation, and Configuration of Ollama

Embedding and LLM model that will be used. This includes Llama3 (LLM) and nomic-embed-text (Embedding Model)

Section 1:
OpenSearch Installation

1. Download the OpenSearch zip using the following link:

   https://artifacts.opensearch.org/releases/bundle/opensearch/2.16.0/opensearch-2.16.0-windows-x64.zip

2. Extract this to a location where you want your OpenSearch to be installed.

3. Set this environment variable in a command prompt:

   set OPENSEARCH_INITIAL_ADMIN_PASSWORD=<*custom-admin-password*>

   This password needs to be a minimum of 8 characters, include one uppercase letter, one lowercase letter, one digit, and one special character.

4. Before running OpenSearch, you must run the following security batch file.

   opensearch-2.16.0\plugins\opensearch-security\tools\install_demo_configuration.bat

5. Test this installation by running the **OPENSEARCH-WINDOWS-INSTALL.BAT** present inside the opensearch-2.16.0 folder. This installs security plugins with basic http authentications.

6. Open a new command prompt and send a request to OpenSearch at port 9200 using the following command (curl is required for Windows or any rest client):

   curl.exe -X GET https://localhost:9200 -u "admin:<*your_password*>" –insecure

   The response should appear as follows:

```
{
    "name" : "hostname-here",
    "cluster_name" : "opensearch",
    "cluster_uuid" : "7Nqtr0LrQTOveFcBb7Kufw",
    "version" : {
        "distribution" : "opensearch",
        "number" : <version>,
        "build_type" : <build-type>,
```

```
        "build_hash" : <build-hash>,
        "build_date" : <build-date>,
        "build_snapshot" : false,
        "lucene_version" : <lucene-version>,
        "minimum_wire_compatibility_version" : "7.10.0",
        "minimum_index_compatibility_version" : "7.0.0"
        },
    "tagline" : "The OpenSearch Project: https://opensearch.org/"
}
```

7. Query the plugin points using the following command

   Note: Here the username and password are both admin, which are the demo user values. When the internal users are configured, these values are changed.

   curl.exe -X GET https://localhost:9200/_cat/plugins?v -u "admin:<*your_password*>" -- insecure

   The response should appear as follows:

```
hostname opensearch-alerting                 2.16.0
hostname opensearch-anomaly-detection        2.16.0
hostname opensearch-asynchronous-search      2.16.0
hostname opensearch-cross-cluster-replication 2.16.0
hostname opensearch-geospatial               2.16.0
hostname opensearch-index-management         2.16.0
hostname opensearch-job-scheduler            2.16.0
hostname opensearch-knn                      2.16.0
hostname opensearch-ml                       2.16.0
hostname opensearch-neural-search            2.16.0
hostname opensearch-notifications            2.16.0
```

```
hostname opensearch-notifications-core        2.16.0

hostname opensearch-observability             2.16.0

hostname opensearch-reports-scheduler         2.16.0

hostname opensearch-security                  2.16.0

hostname opensearch-security-analytics        2.16.0

hostname opensearch-sql                       2.16.0
```

8. Open the **opensearch.yml** file inside the **opensearch-2.16.0/config** folder and add the following for basic initialization of OpenSearch local client:
   Note: These are all default values.

```
cluster.name: <name-of-your-cluster>

#

# ------------------------------------ Node ------------------------------------

#

# Use a descriptive name for the node:

#

node.name: <name-of-your-node>

node.roles: [cluster_manager,data,ingest,search,ml]

node.search.cache.size: 512m

#

# Add custom attributes to the node:

#

# node.attr.rack: r1

#

# ---------------------------------- Paths ------------------------------------

#
```

```
# Path to directory where to store the data (separate multiple locations
by         comma):

#

path.data: path/to/data #add the path where the data for the cluster needs to
be    stored

#

# Path to log files:

#

path.logs: path/to/logs #add the path where the logs for the cluster needs to
be    stored

#

# ---------------------------------- Memory -----------------------------------
---

#

# Lock the memory on startup:

#

bootstrap.memory_lock: true

#

# Make sure that the heap size is set to about half the memory available

# on the system and that the owner of the process is allowed to use this

# limit.

#

# OpenSearch performs poorly when the system is swapping the memory.

#

# ---------------------------------- Network ----------------------------------
---

#
```

```
# Set the bind address to a specific IP (IPv4 or IPv6):

#

network.host: 0.0.0.0

#

# Set a custom port for HTTP:

#

http.port: 9200

#

# For more information, consult the network module documentation.

#

# ---------------------------------- Discovery ----------------------------------

#

# Pass an initial list of hosts to perform discovery when this node is started:

# The default list of hosts is ["127.0.0.1", "[::1]"]

#

discovery.seed_hosts: ["127.0.0.1"]

discovery.type: single-node
```

Note: Do not update anything in the file after **Discovery**.

9. In **path.logs** and **path.data,** provide the paths of locations in which you want the data to be saved.

10. Open **jvm.options** inside the config folder and set the heap sizes to half the memory of machine. For example, if the machine memory is 8gb set the initial and max size to:

    - **Xms4g**
    - **Xmx4g**

Section 2:
OpenSearch Dashboard Installation

1. Download the OpenSearch dashboard from the following link:

   https://artifacts.opensearch.org/releases/bundle/opensearch-dashboards/2.16.0/opensearch-dashboards-2.16.0-windows-x64.zip

2. Run the following command in PowerShell (admin):

   Set-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem LongPathsEnabled -Type DWORD -Value 1 -Force

3. Open the bin folder and run the batch script by double-clicking the **opensearch-dashboards.bat** file. This opens a command prompt with an OpenSearch dashboard instance running. (To stop OpenSearch dashboards, press **Ctrl+C**, or simply close the Command Prompt or Powershell window.)

4. Log in to the OpenSearch dashboard at **localhost:5601** using **admin** as username and the password you created. OpenSearch needs to be running to access the OpenSearch dashboard.

   Note: The admin username and password are demo values, which are default during installation but should be configured by changing the **opensearch-security config** and **internal_users.yml** files.

5. After logging into OpenSearch, navigate to **Dev Tools** and run the following query to create an index-template.

```
PUT _index_template/tc_template
{
  "index_patterns": [
    "*_main"
  ],
  "template": {
    "aliases": {
      "my_logs": {}
    },
    "settings": {"index.knn": true},
            "mappings": {
                "properties": {
                    "id": {
                        "type": "text",
                        "fields": {"keyword": {"type": "keyword",
"ignore_above": 256}}
                    },
                    "metadata": {
```

```
                        "properties": {
                            "owning_file": {
                                "type": "text",
                                "fields": {
                                    "keyword": {"type": "keyword",
"ignore_above": 256}
                                }
                            },
                            "owning_object": {
                                "type": "text",
                                "fields": {
                                    "keyword": {"type": "keyword",
"ignore_above": 256}
                                }
                            },
                            "section_index": {"type": "long"},
                            "last_modified_date": {
                                "type": "text"
                            },
                            "knowledge_bases": {
                                "type": "text",
                                "fields": {
                                    "keyword": {"type": "keyword",
"ignore_above": 256}
                                }
                            },
                            "owning_user": {
                                "type": "text",
                                "fields": {
                                    "keyword": {"type": "keyword",
"ignore_above": 256}
                                }
                            },
                            "owning_group": {
                                "type": "text",
                                "fields": {
                                    "keyword": {"type": "keyword",
"ignore_above": 256}
                                }
                            }
                        }
                    },
                    "text": {
                        "type": "text",
                        "fields": {"keyword": {"type": "keyword",
"ignore_above": 256}}
                    },
                    "vector_field": {
                        "type": "knn_vector",
                        "dimension": 768,
                        "method": {
                            "engine": "nmslib",
```

```
                        "space_type": "cosinesimil",
                        "name": "hnsw",
                        "parameters": {"ef_construction": 512, "m": 16}
                    }
                }
            }
        }
    }
}
```

The name of the index-pattern is your choice. Here, a sample name *_main is given. If you provide an index name with the suffix "_main", then all the applicable fields are filled out automatically when creating your index. Keep in mind that you can provide a prefix or suffix here, just make sure to include that same structure in the index name that you provide.

The **alias** by default is set to my_logs.

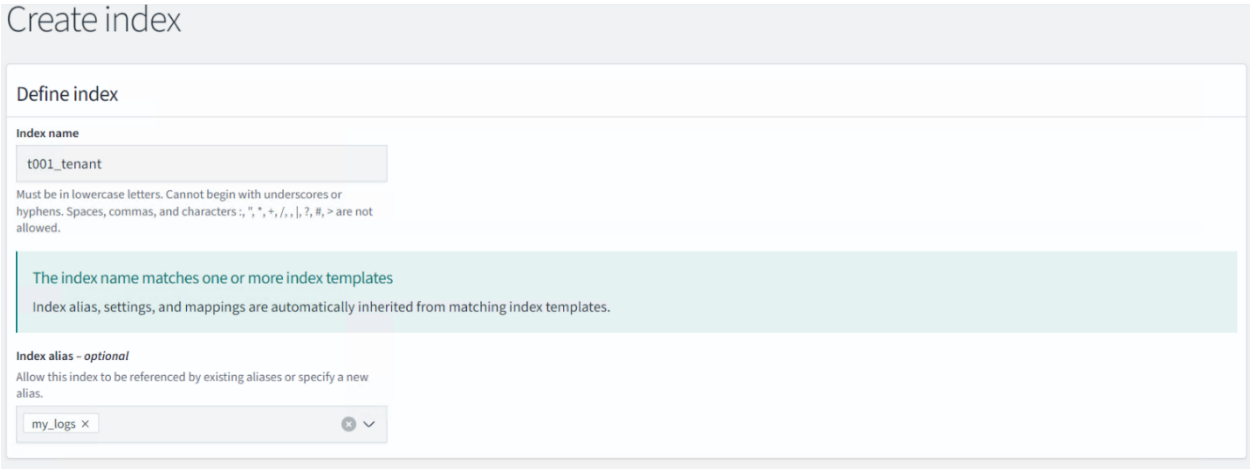
The **name** of the template by default is set to tc_template.

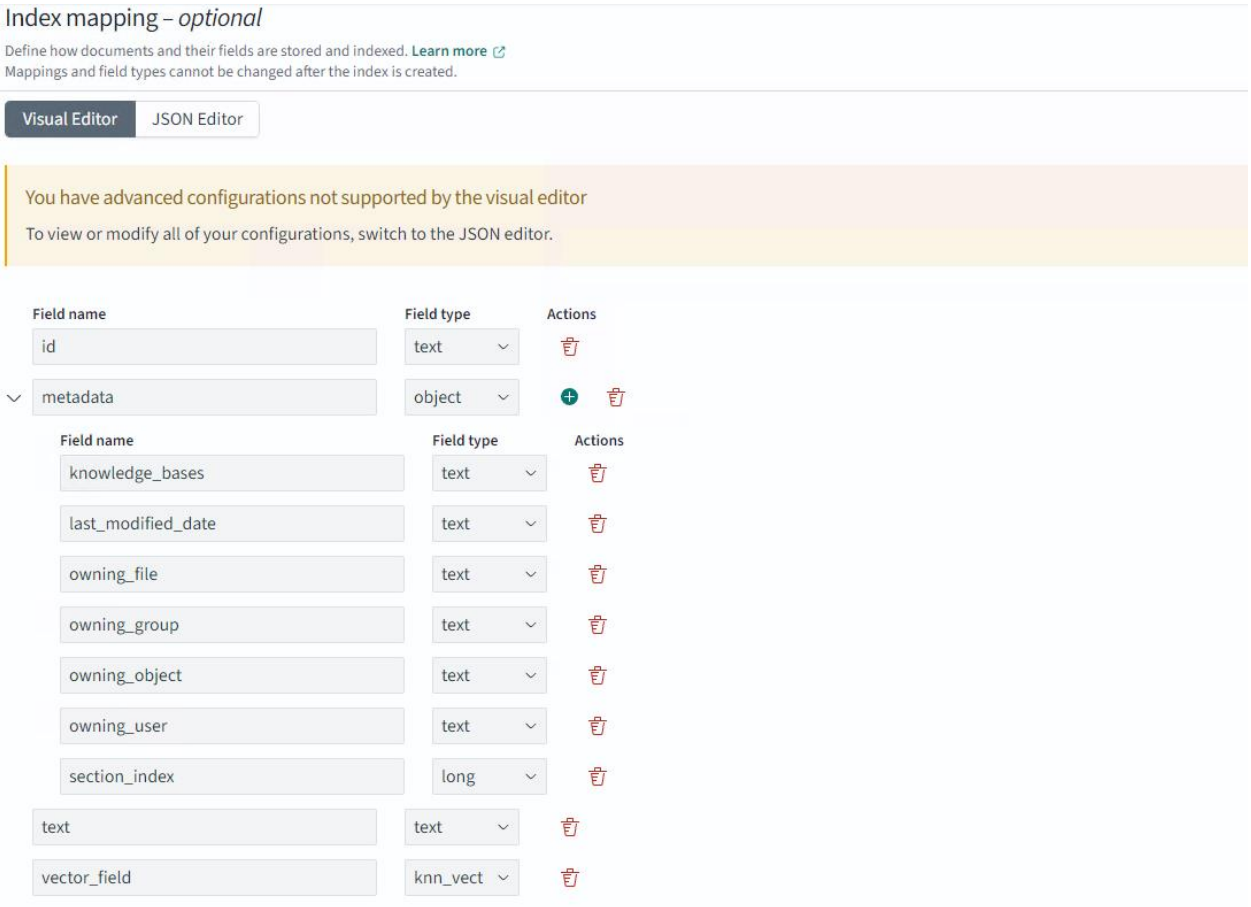
6. After executing this query, a template is created which can be used for index creation. To use the template, navigate to **Index Management** on the OpenSearch dashboard. Click **Indexes** > **Create Index.** Enter the name the index as per the index-pattern provided in the template. Once you do, it will state that the index name matches an index template and will automatically fill out your **Index Mapping** section.

   Notes:

   - If the index pattern is *_main, then the index name should end with _main to apply the template to the index.
   - The template has a custom data field in **knn-vector** so the creation will have to be done using json editor only.

7. For index settings, using one primary shard and no replicas will work for our use case. In **Refresh Interval**, make sure to include seconds with the number. For example, 60s.





8. Once your index is ready, click **Create.**
   The index is created and ready for indexing and searching.

9. Set these environment variables before using OpenSearch:

   NOTE: If you would like to permanently set these environment variables, go into settings to set them. If you do not, you must reset the environment variables every time you open a new command prompt to run this.

   a. **set KNN_LIB_DIR=*<PATH to lib folder inside of opensearch-knn plugin>***

   - Add this path to the PATH variable and add **opensearchknn_common.dll** to the path. That file should also be located inside lib folder of opensearch-knn plugin.

- **NOTE:** If there are too many environment variables in the PATH, it will not pick up opensearchknn_common.dll and OpenSearch will fail. If there are extra environment variables that are not necessary, remove them.

b. **set OPENSEARCH_HOME=<Path to root OpenSearch directory>**

c. **set OPENSEARCH_PATH_CONF=<Path to directory where OpenSearch yml directory is located. Located in config directory>**

d. **set OPENSEARCH_JAVA_HOME=<Path to jdk inside OpenSearch directory>**

10. To run OpenSearch, go to bin folder from within opensearch-2.16.0 folder and run **OPENSEARCH-WINDOWS-INSTALL.BAT**.

    NOTE: If OpenSearch will not start, verify that the environment variables are set properly.

Section 3: Creation/Modification of Users and Roles in OpenSearch

NOTE: This section is optional if you need to modify or create users and roles in OpenSearch.

1. To change the user name and password during creation of the cluster, modification of the **internal_users.yml** is required.

2. After running the **opensearch-windows-install.bat,** certain demo users are created inside the **internal_users.yml.** If new user needs to be added to this, the format is as follows:

   ```
   Change the demo admin role created, change the username to
   desired admin and the hash or password to desired password.
   ```

3. To create the hash of the password, in the command prompt, navigate to **plugins\opensearch-security\tools** and run the following command:

   Hash.bat

4. Insert the password of which you want a hash created.

   a. **Ex**: index@dmin
   b. **reserved**: true- it cannot be changed using the REST API
   c. **hidden**: true- it cannot be returned using REST API
   d. **opendistro security roles**-
   e. **backend_roles-**
   f. **attributes-**
   g. **static**- if static is true then modification will require restart of cluster and modification to **opensearch.yml**

5. To apply the changes made in the **internal_users.yml**, run the following command:

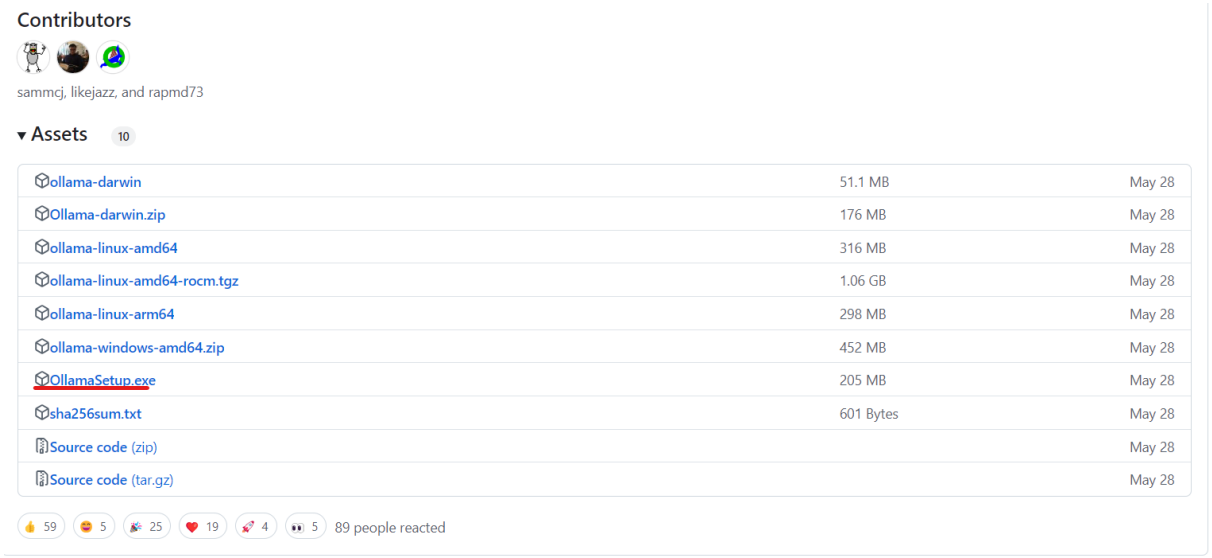
   securityadmin.bat -f ../../../config/opensearch-security/internal_users.yml -t internalusers -icl -nhnv -cacert ../../../config/root-ca.pem -cert ../../../config/kirk.pem -key ../../../config/kirk-key.pem

Section 4: Download, Installation, and Configuration of Ollama

1. Download version v0.1.39 of Ollama from the following link:

   https://github.com/ollama/ollama/releases/tag/v0.1.39

2. Scroll to the bottom and download **OllamaSetup.exe**



3. Once downloaded, install Ollama by running the executable.

4. When installed, Ollama will start. Check your icons for Ollama and stop it from running for configuration.

5. Open a command prompt and set the following environment variables:

   a. **OLLAMA_HOST=0.0.0.0:11434**

   b. **OLLAMA_MAX_LOADED_MODELS=3**

   c. **OLLAMA_KEEP_ALIVE=-1**

d. **OLLAMA_NUM_PARALLEL=10**

6. In the same command prompt, run the following commands:

   a. **ollama pull llama3**

      This pulls the llama3 model, containing a collection of foundation language models.

   b. **Ollama pull nomic-embed-text**

      This pulls the embedding model that you will be using. This must be pulled every time you restart the services.

   c. If you would like to use other models, they can be found at the following link.
   Note: We recommend you use Llama3 models, so use others at your own risk.
   https://ollama.com/library

7. You can also pull your own models with these steps:

   a. Download a GGUF quantized model file from the Hugging Face model library.

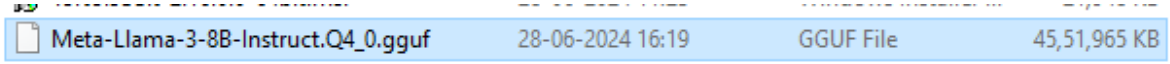
   b. Create an OLLAMA model file with a .modelfile extension.

   c. In this file, include the following line, substituting the path after "FROM" with the actual location of your downloaded GGUF model file. This line specifies the path to your GGUF model file within the OLLAMA model file.

```
FROM C:\Users\Z004Z96E\Downloads\Meta-Llama-3-8B-Instruct-Q4_K_M.gguf
```

For example, let's set up the Llama3Instruct model, which isn't available on the OLLAMA library:

   a. Download the GGUF model file from Hugging Face.



   b. Create a modelfile and import the gguf file location.

```
C: > Users > z004z96e > Desktop > all-mpnet > llama3instruct.modelfile
1    FROM C:\Users\Z004Z96E\Downloads\Meta-Llama-3-8B-Instruct.Q4_0.gguf
```

c. Run OLLAMA from the command line to create the model.use following command :

ollama create model_name -f Modelfile_path

```
C:\Users\z004z96e>ollama create llama3instruct -f C:\Users\z004z96e\Desktop\all-mpnet\llama3instruct.modelfile
transferring model data
using existing layer sha256:1977ae6185ef5bc476e27db85bb3d79ca4bd87e7b03399083c297d9c612d334c
using autodetected template llama3-instruct
creating new layer sha256:04ccc847f217942593fb4d0511fc4f5fa40a431ec7eaef59b84029bb8f821f66
writing manifest
success
```

Now that the model is created, it can be used. You can run it similarly to other models.

```
C:\Users\z004z96e>ollama list
NAME                            ID              SIZE    MODIFIED
llama3instruct:latest           b415f384318d    4.7 GB  About a minute ago
llama2:latest                   78e26419b446    3.8 GB  3 hours ago
nomic-embed-text:latest         0a109f422b47    274 MB  6 hours ago
mxbai-embed-large:latest        468836162de7    669 MB  10 days ago
llava:latest                    8dd30f6b0cb1    4.7 GB  2 weeks ago
llama3:latest                   365c0bd3c000    4.7 GB  2 weeks ago
```

For example: **ollama run llama3instruct**

```
C:\Users\z004z96e>ollama run llama3instruct
>>> hello
Hello! It's nice to meet you. Is there something I can help you with, or would you like to chat?
```

8. Next, we run llama3 and run the Ollama server:

a. **ollama run llama3**

Test to see if it will run properly.

b. **ollama serve**

You can now make API calls with this local server running (port 11434).

Install and Configure Teamcenter AI Services

You must now install Teamcenter AI Microservices and configure the applicable AI features.

For detailed steps regarding the installation of Teamcenter AI Microservices in Deployment Center, please refer to *Teamcenter Installation Using Deployment Center* on Support Center.

For detailed steps regarding the configuration of Teamcenter AI Chat and Natural Language Search, please refer to *Data Indexing and Search Configuration* on Support Center.